```
$Id: extern-tutorial.mm,v 1.3 2018-10-16 14:38:50-07 - - $
PWD: /afs/cats.ucsc.edu/courses/cmps104a-wm/Assignments/extern-tutorial
URL: http://www2.ucsc.edu/courses/cmps104a-wm/:/Assignments/extern-tutorial/
```

This is a short tutorial on the use of the **`extern`** keyword in C and C++. Each brief item comments on a shell command, the output of which is shown after the command. User input is shown in **`Courier-Bold`** and computer output is shown in plain **`Courier`**.

All of these commands are being run on a Unix server. First, let's look at some of the server's properties.

```
-bash-1$ hostname
unix2.lt.ucsc.edu

-bash-2$ uname --kernel-name --kernel-release --kernel-version
Linux 3.10.0-862.14.4.el7.x86_64 #1 SMP Wed Sep 26 15:12:11 UTC 2018

-bash-3$ uname --nodename --operating-system
unix2.lt.ucsc.edu GNU/Linux

-bash-4$ uname --machine --processor --hardware-platform
x86_64 x86_64 x86_64

-bash-5$ which g++
/opt/rh/devtoolset-7/root/usr/bin/g++

-bash-6$ g++ --version
g++ (GCC) 7.3.1 20180303 (Red Hat 7.3.1-5)
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

The program was built with the simple script **`mk`**.

```
-bash-7$ cat -nv code/mk
     1  #!/bin/sh
     2  cid + *.h *.cpp $0
     3  GPPOPT="-g -O0 -std=gnu++17 -Wall -Wextra -Wold-style-cast"
     4  g++ -c $GPPOPT *.cpp
     5  g++ *.o

-bash-8$ cd code; mk
```

Using the command **`file`**(1), we examine the types of the files in the **`code/`** subdirectory.

```
-bash-9$ file code/*
code/HEADER.html: HTML document, ASCII text
code/RCS:         directory
code/a.out:       ELF 64-bit LSB executable, x86-64, version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=cd89bd957b2058cf7efe7d1b8c759657c65ccc7a, not stripped
code/ext.cpp:     C source, ASCII text
code/ext.h:       C source, ASCII text
code/ext.o:       ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV),
not stripped
code/main.cpp:    C source, ASCII text
code/main.o:      ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV),
not stripped
code/mk:          POSIX shell script, ASCII text executable
```

The files in **`code/`** are listed as follows.

```
-bash-10$ ls -goad code/*
-rw------- 1   527 Mar  1  2018 code/HEADER.html
drwx------ 2  2048 Oct 16 14:38 code/RCS
-rwx------ 1 13728 Oct 16 14:38 code/a.out
-rw------- 1   240 Oct 16 14:38 code/ext.cpp
-rw------- 1   199 Oct 16 14:38 code/ext.h
-rw------- 1  6168 Oct 16 14:38 code/ext.o
-rw------- 1   255 Oct 16 14:38 code/main.cpp
-rw------- 1  6824 Oct 16 14:38 code/main.o
-rwx------ 1   118 Oct 16 14:38 code/mk
```

The file **`code/ext.h`** is included in both and links the two. Note the file guards.

```
-bash-11$ cat -nv code/ext.h
     1  // $Id: ext.h,v 1.1 2017-10-11 14:05:24-07 - - $
     2  // This is an example of a header exported by the ext module.
     3
     4  #ifndef __EXT_H__
     5  #define __EXT_H__
     6
     7  extern int ext_var;
     8  void print_ext_var();
     9
    10  #endif
```

The file **`code/ext.cpp`** exports an external variable.

```
-bash-12$ cat -nv code/ext.cpp
     1  // $Id: ext.cpp,v 1.1 2017-10-11 14:05:24-07 - - $
     2  // This is an example of a module exporting an external variable.
     3
     4  #include <stdio.h>
     5
     6  #include "ext.h"
     7
     8  int ext_var = 44;
     9
    10  void print_ext_var() {
    11     printf ("ext_var = %d\n", ext_var);
    12  }
    13
```

The file **`code/main.cpp`** uses an external variable exported from another module.

```
-bash-13$ cat -nv code/main.cpp
     1  // $Id: main.cpp,v 1.1 2017-10-11 14:05:24-07 - - $
     2  // This is an example of a module accessing an external variable.
     3
     4  #include <stdlib.h>
     5
     6  #include "ext.h"
     7
     8  int main() {
     9     print_ext_var();
    10     ext_var = 56;
    11     print_ext_var();
    12     return EXIT_SUCCESS;
    13  }
    14
```

When run, the program produces the following output.

```
-bash-14$ code/a.out
ext_var = 44
ext_var = 56
```

Every module that accesses an external variable must declare it using the **`extern`** keyword. In order to ensure consistency of declaration, this should be placed in a header file. The module exporting the variable, and only that module, then redeclares that same variable without the **`extern`** keyword. Every external variable must be declared without the **`extern`** keyword in exactly one module.

If not declared at all, one gets an undefined external reference error at link time. If declared more than once, then the error is a duplicate declaration error at link time. If not declared as **`extern`** in a header file, then the variables are local to the file and not related.

Now consider the output of running **nm**(1) on each of the object files. The **extern** keyword in the header file marks the variable as external, that is global to both modules. It is redeclared in the file **ext.cpp** without the **extern** keyword, so **nm code/ext.o** produces the following output.

```
-bash-15$ nm code/ext.o
0000000000000000 T _Z13print_ext_varv
0000000000000000 D ext_var
                 U printf
```

On the other hand, running **nm code/main.o** shows that **external_variable** is undefined in that module.

```
-bash-16$ nm code/main.o
                 U _Z13print_ext_varv
                 U ext_var
0000000000000000 T main
```

The sizes of the segments in the object files and executable binary can be obtained via **size**(1).

```
-bash-17$ cd code; size *.o a.out
   text    data     bss     dec     hex filename
    100       4       0     104      68 ext.o
     87       0       0      87      57 main.o
   1325     584       8    1917     77d a.out
```

Looking at the executable image **`a.out`** with **`nm code/a.out`** we see that each symbol has a specific address assigned to it. It also has references included from the library. The letter shows whether the object is Undefined, or belongs to the Text, Data, or BSS segment, or if it is Absolute. See **nm**(1) for a complete explanation.

```
-bash-18$ nm code/a.out
0000000000600df8 d _DYNAMIC
0000000000601000 d _GLOBAL_OFFSET_TABLE_
00000000004005f0 R _IO_stdin_used
0000000000400527 T _Z13print_ext_varv
0000000000400780 r __FRAME_END__
0000000000400610 r __GNU_EH_FRAME_HDR
0000000000601030 D __TMC_END__
0000000000601030 B __bss_start
0000000000601028 D __data_start
00000000004004f0 t __do_global_dtors_aux
0000000000600df0 t __do_global_dtors_aux_fini_array_entry
00000000004005f8 R __dso_handle
0000000000600de8 t __frame_dummy_init_array_entry
                 w __gmon_start__
0000000000600df0 t __init_array_end
0000000000600de8 t __init_array_start
00000000004005e0 T __libc_csu_fini
0000000000400570 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
0000000000601030 D _edata
0000000000601038 B _end
00000000004005e4 T _fini
00000000004003f0 T _init
0000000000400450 T _start
0000000000601030 b completed.6943
0000000000601028 W data_start
0000000000400480 t deregister_tm_clones
000000000060102c D ext_var
0000000000400520 t frame_dummy
0000000000400545 T main
                 U printf@@GLIBC_2.2.5
00000000004004b0 t register_tm_clones
```